Практическое задание №16
(Дважды связанный линейный список)

Создать решение с именем **LinkedList**.

Задание Летучки (см. в директории «Летучки») доработать, реализовав в классе **LinkedList** классы:

- константного итератора const_iterator, возвращающего константную ссылку на связанный с итератором элемент списка,

- реверсивного итератора reverse_iterator (для прохода списка в обратном направлении),

- const_reverse_iterator (константную версию реверсивного итератора reverse_iterator).

В классах итераторов необходимо реализовать следующие методы:

– operator ++ (префиксную и постфиксную формы) для перехода к следующему узлу списка;

– operator -- (префиксную и постфиксную формы) для перехода к предыдущему узлу списка;

– operator * для разыменования итератора; при разыменовании возвращается ссылка на связанный с итератором элемент списка (ссылка на хранящееся в узле значение _value);

– operator -> - селектор, возвращающий указатель на связанный с итератором элемент списка (указатель на хранящееся в узле значение _value);

– operator == сравнения итераторов на равенство;

– operator != сравнения итераторов на неравенство.

В классе LinkedList необходимо дополнительно реализовать:

1. **Метод const_iterator begin() const, возвращающий const_iterator, соответствующий первому элементу списка.**
2. **Метод reverse_iterator rbegin(), возвращающий итератор, соответствующий последнему элементу списка.**
3. **Метод const_reverse_iterator rbegin() const, возвращающий итератор, соответствующий последнему элементу списка.**
4. **Метод const_iterator end() const, возвращающий const_iterator, соответствующий фиктивному элементу, следующему за последним элементом списка.**
5. **Метод reverse_iterator rend(), возвращающий reverse_iterator, соответствующий фиктивному элементу, находящемуся перед первым элементом списка.**

6. **Метод const_reverse_iterator rend() const, возвращающий const_reverse_iterator, соответствующий фиктивному элементу, находящемуся перед первым элементом списка.**

Реализовать тесты для разработанных методов класса. В тестах должна быть продемонстрирована возможность прохода по списку в прямом (от начала к концу) и в обратном (от конца к началу) направлениях (с использованием итераторов всех четырёх классов), а также применение к LinkedList различных алгоритмов STL.

Для сдачи проекта использовать структуру в файловой системе:

- gxxxxx/16/ForwardList.h – файл с шаблонным классом;
- gxxxxx/16/ForwardList.Tests.cpp – файлы с тестами пунктов задания (имена файлов давать в соответствии с вариантами).